Verzeo Minor Project-Mobile Price Prediction

Data Import

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data=pd.read_csv('mobile_price_range_data.csv')
```

In [3]:
```
data
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	px_width	ram	sc_h	sc_w	talk_time	t
0	842	0	2.2	0	1	0	7	0.6	188	2	...	20	756	2549	9	7	19	
1	1021	1	0.5	1	0	1	53	0.7	136	3	...	905	1988	2631	17	3	7	
2	563	1	0.5	1	2	1	41	0.9	145	5	...	1263	1716	2603	11	2	9	
3	615	1	2.5	0	0	0	10	0.8	131	6	...	1216	1786	2769	16	8	11	
4	1821	1	1.2	0	13	1	44	0.6	141	2	...	1208	1212	1411	8	2	15	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1995	794	1	0.5	1	0	1	2	0.8	106	6	...	1222	1890	668	13	4	19	
1996	1965	1	2.6	1	0	0	39	0.2	187	4	...	915	1965	2032	11	10	16	
1997	1911	0	0.9	1	1	1	36	0.7	108	8	...	868	1632	3057	9	1	5	
1998	1512	0	0.9	0	4	1	46	0.1	145	5	...	336	670	869	18	10	19	
1999	510	1	2.0	1	5	1	45	0.9	168	6	...	483	754	3919	19	4	2	

2000 rows × 21 columns

Data Analysis and cleaning

In [4]:
```
data.shape
```

Out[4]:
```
(2000, 21)
```

Checking for null values

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

In [7]:
```
data.describe()
```

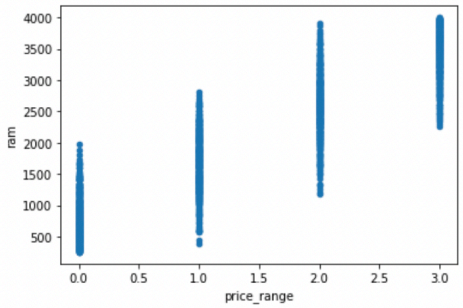
Out[7]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	...	px_height	
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	20
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	0.501750	140.249000	4.520500	...	645.108000	12
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	0.288416	35.399655	2.287837	...	443.780811	4
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	0.100000	80.000000	1.000000	...	0.000000	5
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	0.200000	109.000000	3.000000	...	282.750000	8
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	0.500000	141.000000	4.000000	...	564.000000	12
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	0.800000	170.000000	7.000000	...	947.250000	16
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	1.000000	200.000000	8.000000	...	1960.000000	19

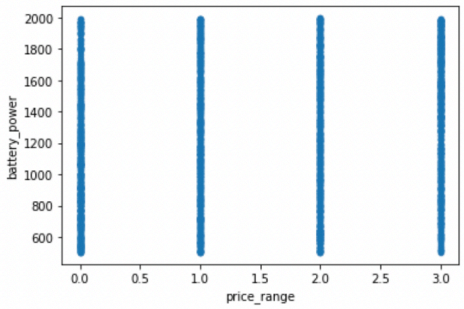
8 rows × 21 columns

Ploting datas to analyse the trends

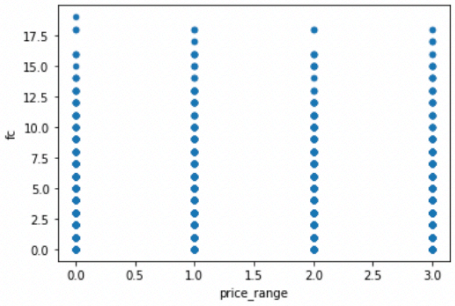
In [8]:
```
data.plot(x='price_range',y='ram',kind='scatter')
plt.show()
```



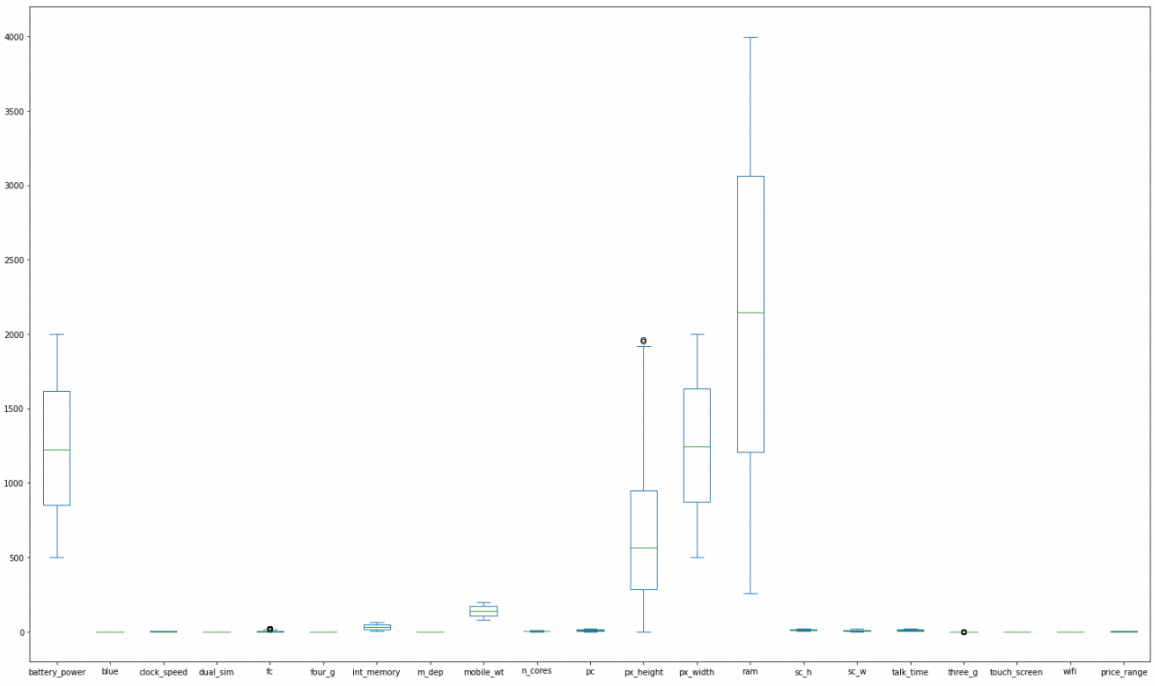
In [9]:
```
data.plot(x='price_range',y='battery_power',kind='scatter')
plt.show()
```



In [10]:
```
data.plot(x='price_range',y='fc',kind='scatter')
plt.show()
```



In [11]:
```
data.plot(kind='box',figsize=(25,15))
plt.show()
```



Spliting into Train and Test set

In [12]:
```
X=data.drop('price_range',axis=1)
X
```

Out[12]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time
0	842	0	2.2	0	1	0	7	0.6	188	2	2	20	756	2549	9	7	19
1	1021	1	0.5	1	0	1	53	0.7	136	3	6	905	1988	2631	17	3	7
2	563	1	0.5	1	2	1	41	0.9	145	5	6	1263	1716	2603	11	2	9
3	615	1	2.5	0	0	0	10	0.8	131	6	9	1216	1786	2769	16	8	11
4	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208	1212	1411	8	2	15
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1995	794	1	0.5	1	0	1	2	0.8	106	6	14	1222	1890	668	13	4	19
1996	1965	1	2.6	1	0	0	39	0.2	187	4	3	915	1965	2032	11	10	16
1997	1911	0	0.9	1	1	1	36	0.7	108	8	3	868	1632	3057	9	1	5
1998	1512	0	0.9	0	4	1	46	0.1	145	5	5	336	670	869	18	10	19
1999	510	1	2.0	1	5	1	45	0.9	168	6	16	483	754	3919	19	4	2

2000 rows × 20 columns

In [13]:
```
Y=data['price_range']
Y
```

Out[13]:
```
0       1
1       2
2       2
3       2
4       1
       ..
1995    0
1996    2
1997    3
1998    0
1999    3
Name: price_range, Length: 2000, dtype: int64
```

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,random_state=1)
```

In [16]:
```
X_train
```

Out[16]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height	px_width	ram	sc_h	sc_w	talk_time
1194	1299	0	2.8	1	2	0	60	0.8	187	8	6	1333	1838	1412	13	1	20
45	1514	0	2.9	0	0	0	27	0.2	118	3	1	186	1810	1152	8	3	20
1477	1150	1	2.7	0	0	0	39	0.4	151	8	1	397	800	999	6	2	11
1293	1702	0	1.0	0	7	0	36	0.1	177	5	9	1240	1931	1430	8	6	10
1736	1779	1	2.6	1	2	0	5	0.8	90	1	3	1225	1717	1246	12	2	20
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1791	1203	1	0.5	1	0	1	11	0.9	109	2	12	35	510	1672	17	13	19
1096	1154	0	2.0	0	6	1	35	0.8	159	5	16	1003	1827	3262	16	15	16
1932	718	1	1.6	1	1	1	30	0.4	93	3	12	662	997	1601	10	9	12
235	1523	1	1.8	0	6	1	11	0.1	129	1	8	148	1606	707	19	8	19
1061	1522	1	0.7	1	4	0	28	0.2	124	1	5	952	1191	1368	14	5	17

1400 rows × 20 columns

In [17]:
```
Y_train
```

Out[17]:
```
1194    1
45      1
1477    0
1293    2
1736    1
       ..
1791    0
1096    3
1932    1
235     0
1061    1
Name: price_range, Length: 1400, dtype: int64
```

Now we have split the data into train and test set where train set contains 1400 data and test 600 data.We first run models on x_train and y_train then make the model to predict price for x_test and then compare it with y_test to find the accuracy....

1) Logistic Regression

In [18]:
```
#### using Standard Scaler ...
```

In [19]:
```
from sklearn.preprocessing import StandardScaler
std=StandardScaler()

X_train_std=std.fit_transform(X_train)

X_test_std=std.transform(X_test)
```

In [20]:
```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train_std,Y_train)
```

Out[20]:
```
LogisticRegression()
```

In [21]:
```
Y_pred=lr.predict(X_test_std)

Y_pred
```

Out[21]:
```
array([0, 0, 1, 1, 3, 2, 0, 2, 2, 3, 0, 3, 1, 1, 3, 0, 0, 1, 1, 1, 3, 3,
       1, 2, 3, 2, 2, 3, 2, 2, 2, 1, 2, 0, 3, 3, 0, 0, 0, 0, 2, 1, 2, 1,
       0, 1, 2, 2, 1, 2, 1, 3, 1, 3, 1, 3, 1, 2, 3, 1, 0, 2, 0, 3, 2, 1,
       1, 2, 3, 2, 1, 1, 0, 3, 3, 1, 2, 1, 0, 0, 0, 3, 1, 2, 3, 2, 2, 0,
       1, 1, 3, 0, 1, 1, 2, 3, 3, 0, 3, 3, 3, 3, 0, 1, 2, 0, 0, 1, 0, 2,
       0, 3, 1, 1, 2, 2, 3, 1, 2, 1, 2, 0, 0, 0, 3, 0, 1, 1, 0, 1, 0, 2,
       0, 3, 3, 0, 3, 2, 2, 1, 0, 0, 3, 1, 0, 2, 0, 0, 0, 1, 3, 3, 2, 1,
       0, 2, 1, 0, 3, 1, 3, 1, 2, 2, 3, 1, 2, 2, 3, 2, 2, 0, 2, 2, 0, 3,
       3, 0, 0, 3, 0, 3, 0, 0, 2, 3, 3, 1, 2, 1, 2, 3, 2, 2, 0, 1, 2, 3,
       3, 1, 3, 2, 0, 1, 2, 2, 2, 1, 0, 2, 0, 3, 2, 0, 2, 0, 1, 0, 2, 1,
       0, 1, 2, 1, 3, 3, 0, 3, 2, 1, 0, 2, 3, 3, 2, 1, 0, 1, 3, 2, 3, 2,
       1, 1, 0, 1, 3, 0, 3, 3, 3, 1, 2, 1, 0, 1, 1, 3, 2, 0, 3, 1, 1, 1,
       2, 1, 2, 0, 3, 2, 2, 0, 0, 1, 3, 3, 0, 1, 1, 2, 3, 2, 1, 2, 2, 0,
       0, 3, 0, 0, 1, 3, 2, 3, 0, 0, 1, 2, 3, 2, 2, 2, 0, 0, 3, 1, 1, 0,
       0, 0, 3, 1, 2, 1, 0, 2, 3, 2, 3, 3, 3, 1, 1, 1, 2, 1, 1, 0, 3, 3,
       3, 1, 1, 0, 1, 2, 3, 2, 0, 3, 3, 2, 3, 2, 1, 2, 3, 2, 3, 2, 1, 3,
       1, 0, 3, 1, 3, 2, 2, 2, 3, 0, 2, 0, 1, 1, 2, 3, 0, 0, 2, 2, 1, 0,
       3, 3, 3, 0, 2, 2, 2, 3, 3, 3, 3, 0, 3, 2, 0, 1, 3, 2, 2, 0, 2, 2,
       3, 3, 1, 0, 1, 0, 0, 0, 1, 1, 3, 1, 1, 3, 0, 1, 1, 3, 3, 2, 2, 3,
       3, 3, 3, 3, 0, 1, 3, 1, 2, 2, 1, 1, 1, 0, 2, 2, 1, 1, 0, 0, 3, 2,
       0, 3, 3, 2, 1, 3, 0, 2, 3, 0, 2, 3, 1, 0, 2, 0, 2, 0, 3, 2, 0, 1,
       0, 2, 0, 3, 2, 2, 0, 1, 0, 1, 1, 2, 1, 2, 1, 0, 1, 2, 0, 2, 0, 1,
       3, 1, 3, 2, 2, 0, 2, 1, 2, 1, 2, 2, 0, 0, 2, 0, 2, 1, 1, 2, 1, 3,
       2, 3, 0, 3, 1, 1, 0, 0, 1, 0, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 0, 3,
       3, 1, 2, 1, 2, 0, 1, 2, 3, 3, 1, 2, 0, 1, 1, 0, 0, 2, 3, 2, 1, 2,
       3, 0, 1, 3, 1, 2, 2, 0, 3, 1, 2, 0, 2, 3, 3, 2, 2, 3, 3, 2, 0, 1,
       0, 2, 3, 2, 3, 0, 2, 3, 1, 0, 2, 2, 0, 2, 2, 2, 1, 3, 2, 2, 3, 0,
       2, 0, 2, 2, 3, 3])
```

In [22]:
```
from sklearn.metrics import accuracy_score
lr_ac=accuracy_score(Y_test,Y_pred)

lr_ac
```

Out[22]:
```
0.935
```

In [23]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_pred,Y_test)
```

Out[23]:
```
array([[133,   6,   0,   0],
       [  2, 137,   8,   0],
       [  0,   6, 150,   7],
       [  0,   0,  10, 141]])
```

2) KNN Classification

In [24]:
```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train_std,Y_train)
```

Out[24]:
```
KNeighborsClassifier()
```

In [25]:
```
Y_pred=knn.predict(X_test_std)
Y_pred
```

Out[25]:
```
array([0, 1, 1, 0, 1, 1, 0, 2, 1, 3, 0, 3, 2, 0, 2, 0, 1, 1, 1, 2, 1, 2,
       1, 3, 3, 2, 2, 1, 1, 3, 1, 1, 2, 0, 3, 1, 0, 0, 1, 1, 0, 1, 2, 0,
       0, 0, 1, 2, 0, 2, 0, 1, 0, 2, 1, 3, 0, 2, 2, 0, 0, 3, 0, 2, 0, 2,
       1, 2, 3, 2, 1, 3, 0, 3, 2, 1, 3, 0, 1, 0, 0, 3, 0, 1, 3, 2, 2, 0,
       2, 0, 1, 0, 0, 1, 0, 2, 2, 0, 1, 1, 2, 3, 0, 0, 2, 0, 0, 1, 1, 0,
       0, 3, 2, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 3, 1, 1, 0, 1, 1, 0, 2,
       0, 2, 2, 1, 2, 2, 1, 2, 1, 1, 3, 0, 0, 2, 0, 1, 0, 0, 2, 3, 1, 1,
       0, 0, 1, 0, 2, 1, 2, 0, 2, 2, 3, 0, 3, 2, 3, 1, 1, 0, 1, 0, 1, 3,
       2, 0, 0, 3, 0, 3, 0, 2, 1, 3, 3, 1, 1, 1, 2, 1, 2, 3, 0, 2, 1, 3,
       2, 0, 3, 2, 0, 3, 2, 0, 1, 0, 0, 1, 1, 2, 2, 0, 1, 0, 1, 0, 3, 1,
       2, 2, 2, 2, 2, 1, 0, 3, 2, 1, 0, 1, 3, 3, 1, 1, 0, 1, 2, 0, 3, 2,
       0, 1, 0, 2, 1, 0, 2, 2, 2, 0, 0, 1, 1, 1, 1, 2, 1, 0, 1, 0, 1, 2,
       2, 2, 2, 0, 1, 0, 3, 0, 1, 2, 2, 3, 0, 1, 1, 2, 3, 1, 1, 0, 2, 1,
       1, 2, 0, 0, 1, 3, 1, 3, 1, 0, 1, 3, 2, 3, 0, 3, 0, 1, 1, 0, 3, 0,
       1, 0, 1, 1, 2, 0, 0, 1, 3, 2, 2, 2, 3, 0, 2, 1, 2, 0, 0, 0, 3, 2,
       3, 0, 1, 0, 1, 2, 1, 2, 2, 3, 3, 0, 2, 2, 0, 3, 2, 3, 3, 3, 1, 3,
       0, 0, 1, 2, 2, 2, 1, 1, 2, 1, 2, 0, 1, 0, 1, 3, 0, 1, 1, 1, 2, 1,
       1, 3, 2, 0, 3, 1, 1, 2, 3, 3, 3, 0, 3, 0, 0, 2, 3, 2, 3, 0, 1, 2,
       3, 3, 2, 0, 0, 0, 0, 0, 1, 1, 2, 1, 3, 1, 0, 1, 1, 3, 2, 1, 3, 3,
       3, 2, 1, 2, 0, 1, 2, 0, 2, 3, 1, 1, 0, 0, 2, 0, 0, 1, 1, 1, 3, 1,
       1, 3, 3, 3, 0, 3, 0, 2, 2, 0, 0, 1, 1, 0, 1, 0, 2, 1, 3, 2, 0, 0,
       0, 1, 0, 3, 2, 2, 0, 2, 0, 3, 1, 1, 1, 3, 2, 2, 1, 1, 0, 1, 2, 0,
       3, 2, 3, 1, 1, 2, 1, 2, 2, 0, 1, 3, 1, 0, 3, 0, 0, 0, 0, 2, 1, 2,
       2, 2, 0, 3, 0, 0, 0, 0, 2, 0, 2, 0, 3, 0, 2, 3, 1, 2, 3, 1, 1, 3,
       3, 1, 2, 1, 2, 0, 0, 2, 2, 3, 0, 0, 1, 1, 0, 0, 0, 3, 3, 0, 0, 3,
       3, 0, 0, 2, 1, 1, 0, 0, 1, 0, 0, 0, 1, 2, 1, 1, 3, 1, 1, 1, 1, 1,
       1, 2, 3, 3, 2, 0, 2, 3, 0, 0, 1, 1, 0, 1, 1, 3, 0, 3, 3, 2, 3, 0,
       0, 0, 1, 2, 3, 3])
```

In [26]:
```
knn_ac=accuracy_score(Y_test,Y_pred)
knn_ac
```

Out[26]:
```
0.48833333333333334
```

In [27]:
```
confusion_matrix(Y_pred,Y_test)
```

Out[27]:
```
array([[98, 58, 23,  0],
       [33, 60, 57, 26],
       [ 4, 27, 60, 47],
       [ 0,  4, 28, 75]])
```

SVM Classifier with linear and rbf kernel

In [28]:
```
from sklearn.svm import SVC
linear=SVC(kernel="linear")
linear.fit(X_train,Y_train)
```

Out[28]:
```
SVC(kernel='linear')
```

In [29]:
```
Y_pred = linear.predict(X_test)
Y_pred
```

Out[29]:
```
array([0, 0, 1, 1, 2, 2, 0, 2, 2, 3, 0, 3, 1, 1, 3, 0, 0, 1, 1, 1, 3, 3,
       1, 2, 3, 2, 2, 3, 2, 2, 2, 1, 2, 0, 3, 3, 0, 0, 0, 1, 2, 1, 2, 1,
       0, 1, 2, 2, 1, 2, 1, 3, 1, 3, 1, 3, 1, 3, 3, 1, 0, 2, 1, 3, 2, 1,
       1, 2, 3, 2, 1, 2, 0, 3, 3, 1, 2, 1, 0, 0, 0, 3, 1, 2, 3, 2, 2, 0,
       1, 1, 3, 0, 1, 1, 2, 3, 3, 0, 3, 3, 3, 3, 0, 1, 2, 0, 0, 1, 0, 2,
       0, 3, 1, 1, 2, 2, 3, 1, 2, 1, 1, 0, 0, 0, 3, 0, 1, 1, 0, 1, 0, 2,
       0, 3, 3, 0, 3, 2, 2, 1, 0, 0, 3, 1, 0, 2, 0, 0, 0, 1, 3, 3, 1, 1,
       0, 1, 1, 0, 3, 2, 3, 1, 2, 2, 3, 1, 2, 2, 3, 2, 2, 0, 2, 2, 0, 3,
       3, 0, 0, 3, 0, 3, 0, 0, 2, 3, 3, 1, 2, 1, 2, 3, 2, 2, 0, 1, 1, 3,
       3, 1, 3, 2, 0, 1, 2, 2, 1, 1, 0, 2, 0, 3, 2, 0, 3, 0, 1, 0, 2, 1,
       0, 1, 2, 2, 3, 3, 0, 3, 2, 1, 0, 2, 3, 3, 2, 1, 0, 1, 3, 2, 3, 2,
       2, 2, 0, 1, 3, 0, 3, 3, 3, 1, 2, 2, 0, 1, 1, 3, 2, 0, 3, 1, 1, 1,
       2, 1, 2, 0, 3, 2, 2, 0, 0, 1, 3, 3, 0, 1, 1, 2, 3, 2, 2, 2, 2, 0,
       0, 3, 0, 0, 1, 3, 2, 3, 0, 0, 1, 2, 3, 2, 2, 2, 0, 0, 3, 1, 1, 0,
       0, 0, 3, 1, 2, 1, 0, 2, 3, 2, 3, 3, 3, 1, 1, 1, 2, 1, 1, 0, 3, 3,
       3, 1, 1, 0, 1, 2, 3, 2, 0, 3, 3, 2, 2, 2, 2, 3, 2, 3, 2, 1, 3,
       1, 0, 3, 1, 3, 1, 2, 2, 3, 0, 2, 0, 1, 1, 2, 3, 0, 0, 2, 3, 1, 0,
       3, 3, 3, 0, 2, 2, 2, 3, 3, 3, 3, 0, 3, 2, 0, 1, 3, 2, 2, 0, 2, 2,
       3, 3, 1, 0, 1, 0, 0, 0, 1, 2, 3, 1, 1, 3, 0, 1, 1, 3, 3, 2, 2, 3,
       3, 3, 2, 2, 0, 1, 3, 1, 1, 2, 1, 1, 1, 0, 2, 1, 1, 1, 0, 0, 3, 2,
       0, 3, 3, 1, 1, 3, 0, 2, 3, 0, 2, 3, 1, 0, 2, 0, 2, 0, 3, 2, 0, 1,
       0, 2, 0, 3, 2, 2, 0, 1, 0, 1, 1, 2, 1, 2, 1, 1, 1, 2, 0, 2, 1, 1,
       3, 1, 1, 3, 2, 2, 0, 2, 2, 2, 1, 1, 2, 0, 0, 2, 0, 2, 1, 1, 2, 2,
       2, 3, 0, 3, 1, 1, 0, 0, 1, 0, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 0, 3,
       3, 1, 2, 1, 2, 0, 1, 2, 3, 3, 1, 2, 0, 1, 1, 0, 0, 2, 3, 2, 1, 2,
       3, 1, 1, 3, 1, 2, 2, 0, 3, 1, 2, 0, 2, 3, 3, 2, 2, 3, 3, 2, 0, 1,
       0, 2, 3, 2, 3, 0, 2, 3, 1, 0, 2, 2, 0, 1, 2, 2, 0, 3, 3, 2, 3, 0,
       2, 0, 2, 2, 3, 3])
```

In [30]:
```
linear_ac=accuracy_score(Y_test, Y_pred)
linear_ac
```

Out[30]:
```
0.965
```

In [31]:
```
confusion_matrix(Y_pred,Y_test)
```

Out[31]:
```
array([[133,   2,   0,   0],
       [  2, 144,   6,   0],
       [  0,   3, 157,   3],
       [  0,   0,   5, 145]])
```

In [32]:
```
rbf=SVC(kernel="rbf")
rbf.fit(X_train,Y_train)
Y_pred = rbf.predict(X_test)
Y_pred
```

Out[32]:
```
array([0, 0, 1, 0, 3, 2, 0, 2, 2, 3, 0, 3, 1, 1, 3, 0, 0, 1, 1, 1, 3, 3,
       1, 2, 3, 2, 2, 3, 2, 2, 2, 1, 2, 0, 3, 3, 0, 0, 0, 1, 1, 1, 2, 1,
       0, 1, 2, 2, 1, 2, 1, 3, 1, 3, 1, 3, 3, 1, 0, 2, 1, 3, 2, 1,
       0, 2, 3, 2, 1, 1, 0, 3, 3, 1, 2, 1, 0, 0, 0, 3, 1, 2, 3, 2, 2, 0,
       1, 1, 3, 0, 1, 1, 2, 3, 3, 0, 2, 3, 3, 3, 0, 1, 2, 0, 0, 1, 0, 2,
       0, 3, 1, 1, 2, 2, 3, 1, 2, 1, 2, 0, 0, 0, 3, 0, 1, 1, 0, 1, 0, 3,
       0, 3, 3, 0, 3, 2, 2, 1, 0, 0, 3, 1, 0, 2, 0, 0, 0, 1, 3, 3, 1, 1,
       0, 2, 1, 0, 3, 1, 3, 1, 3, 2, 3, 1, 2, 1, 3, 2, 2, 0, 2, 2, 0, 3,
       3, 0, 0, 3, 0, 3, 0, 0, 2, 3, 3, 1, 2, 1, 2, 3, 2, 3, 0, 1, 2, 3,
       3, 1, 3, 2, 0, 1, 2, 2, 2, 1, 1, 2, 0, 2, 2, 0, 2, 0, 1, 0, 2, 1,
       0, 1, 2, 1, 3, 3, 0, 3, 2, 1, 0, 2, 3, 3, 2, 1, 0, 1, 3, 2, 3, 2,
       2, 1, 0, 1, 2, 0, 3, 3, 3, 1, 2, 1, 0, 1, 1, 3, 2, 0, 3, 1, 1, 1,
       2, 1, 2, 0, 3, 2, 2, 0, 0, 1, 3, 3, 0, 1, 1, 2, 3, 2, 2, 2, 2, 0,
       0, 3, 0, 0, 1, 3, 2, 3, 0, 0, 1, 3, 3, 3, 1, 2, 0, 0, 3, 1, 1, 0,
       0, 0, 3, 1, 2, 1, 0, 2, 3, 1, 3, 3, 3, 1, 1, 1, 2, 1, 1, 0, 3, 3,
       3, 1, 1, 0, 1, 2, 3, 2, 0, 3, 3, 2, 2, 2, 1, 2, 3, 2, 3, 2, 0, 3,
       1, 0, 3, 1, 3, 2, 2, 2, 3, 0, 2, 0, 1, 1, 2, 3, 0, 0, 2, 3, 1, 0,
       3, 3, 3, 0, 2, 2, 1, 3, 3, 3, 3, 0, 3, 2, 0, 1, 3, 2, 2, 0, 2, 2,
       3, 3, 1, 0, 1, 0, 0, 0, 0, 2, 3, 1, 1, 3, 0, 1, 1, 3, 3, 2, 2, 3,
       3, 3, 2, 3, 0, 1, 3, 1, 1, 2, 1, 1, 1, 0, 2, 2, 1, 1, 0, 0, 2, 2,
       0, 3, 3, 2, 1, 3, 0, 2, 3, 0, 2, 3, 1, 0, 2, 0, 2, 0, 3, 2, 0, 1,
       0, 2, 0, 3, 2, 2, 0, 1, 0, 2, 1, 2, 1, 2, 1, 0, 1, 2, 0, 2, 1, 1,
       3, 1, 3, 2, 2, 0, 2, 1, 2, 1, 1, 2, 0, 0, 2, 0, 2, 1, 1, 2, 1, 2,
       2, 3, 0, 3, 1, 1, 0, 0, 1, 0, 3, 0, 3, 1, 1, 3, 3, 3, 3, 1, 0, 3,
       3, 1, 2, 1, 2, 0, 1, 2, 3, 2, 1, 2, 0, 1, 1, 0, 0, 2, 3, 2, 1, 2,
       3, 0, 1, 3, 1, 2, 2, 0, 3, 1, 2, 0, 2, 3, 3, 2, 2, 3, 2, 2, 0, 1,
       0, 2, 3, 2, 3, 0, 2, 3, 1, 0, 2, 2, 0, 2, 2, 2, 0, 3, 3, 2, 3, 0,
       2, 0, 2, 2, 3, 3])
```

In [33]:
```
rbf_ac=accuracy_score(Y_test, Y_pred)
rbf_ac
```

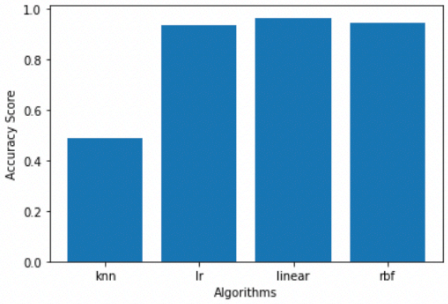
Out[33]:
```
0.9433333333333334
```

In [34]:
```
confusion_matrix(Y_pred,Y_test)
```

Out[34]:
```
array([[134,   7,   0,   0],
       [  1, 138,  10,   0],
       [  0,   4, 151,   5],
       [  0,   0,   7, 143]])
```

Classification report for Models.

In [35]:
```
plt.bar(x=['knn','lr','linear','rbf'],height=[knn_ac,lr_ac,linear_ac,rbf_ac])
plt.xlabel("Algorithms")
plt.ylabel("Accuracy Score")
plt.show()
```



From here we understand that best model for our data set is SVM Classifier with linear kernel

In [36]:
```
linear.predict(X_test)
```

Out[36]:
```
array([0, 0, 1, 1, 2, 2, 0, 2, 2, 3, 0, 3, 1, 1, 3, 0, 0, 1, 1, 1, 3, 3,
       1, 2, 3, 2, 2, 3, 2, 2, 2, 1, 2, 0, 3, 3, 0, 0, 0, 1, 2, 1, 2, 1,
       0, 1, 2, 2, 1, 2, 1, 3, 1, 3, 1, 3, 1, 3, 3, 1, 0, 2, 1, 3, 2, 1,
       1, 2, 3, 2, 1, 2, 0, 3, 3, 1, 2, 1, 0, 0, 0, 3, 1, 2, 3, 2, 2, 0,
       1, 1, 3, 0, 1, 1, 2, 3, 3, 0, 3, 3, 3, 3, 0, 1, 2, 0, 0, 1, 0, 2,
       0, 3, 1, 1, 2, 2, 3, 1, 2, 1, 1, 0, 0, 0, 3, 0, 1, 1, 0, 1, 0, 2,
       0, 3, 3, 0, 3, 2, 2, 1, 0, 0, 3, 1, 0, 2, 0, 0, 0, 1, 3, 3, 1, 1,
       0, 1, 1, 0, 3, 2, 3, 1, 2, 2, 3, 1, 2, 2, 3, 2, 2, 0, 2, 2, 0, 3,
       3, 0, 0, 3, 0, 3, 0, 0, 2, 3, 3, 1, 2, 1, 2, 3, 2, 2, 0, 1, 1, 3,
       3, 1, 3, 2, 0, 1, 2, 2, 1, 1, 0, 2, 0, 3, 2, 0, 3, 0, 1, 0, 2, 1,
       0, 1, 2, 2, 3, 3, 0, 3, 2, 1, 0, 2, 3, 3, 2, 1, 0, 1, 3, 2, 3, 2,
       2, 2, 0, 1, 3, 0, 3, 3, 3, 1, 2, 2, 0, 1, 1, 3, 2, 0, 3, 1, 1, 1,
       2, 1, 2, 0, 3, 2, 2, 0, 0, 1, 3, 3, 0, 1, 1, 2, 3, 2, 2, 2, 2, 0,
       0, 3, 0, 0, 1, 3, 2, 3, 0, 0, 1, 2, 3, 2, 2, 2, 0, 0, 3, 1, 1, 0,
       0, 0, 3, 1, 2, 1, 0, 2, 3, 2, 3, 3, 3, 1, 1, 1, 2, 1, 1, 0, 3, 3,
       3, 1, 1, 0, 1, 2, 3, 2, 0, 3, 3, 2, 2, 2, 2, 2, 3, 2, 3, 2, 1, 3,
       1, 0, 3, 1, 3, 1, 2, 2, 3, 0, 2, 0, 1, 1, 2, 3, 0, 0, 2, 3, 1, 0,
       3, 3, 3, 0, 2, 2, 2, 3, 3, 3, 3, 0, 3, 2, 0, 1, 3, 2, 2, 0, 2, 2,
       3, 3, 1, 0, 1, 0, 0, 0, 1, 2, 3, 1, 1, 3, 0, 1, 1, 3, 3, 2, 2, 3,
       3, 3, 2, 2, 0, 1, 3, 1, 1, 2, 1, 1, 1, 0, 2, 1, 1, 1, 0, 0, 3, 2,
       0, 3, 3, 1, 1, 3, 0, 2, 3, 0, 2, 3, 1, 0, 2, 0, 2, 0, 3, 2, 0, 1,
       0, 2, 0, 3, 2, 2, 0, 1, 0, 1, 1, 2, 1, 2, 1, 1, 1, 2, 0, 2, 1, 1,
       3, 1, 3, 2, 2, 0, 2, 2, 2, 1, 1, 2, 0, 0, 2, 0, 2, 1, 1, 2, 1, 2,
       2, 3, 0, 3, 1, 1, 0, 0, 1, 0, 3, 1, 3, 1, 1, 3, 3, 3, 3, 1, 0, 3,
       3, 1, 2, 1, 2, 0, 1, 2, 3, 3, 1, 2, 0, 1, 1, 0, 0, 2, 3, 2, 1, 2,
       3, 1, 1, 3, 1, 2, 2, 0, 3, 1, 2, 0, 2, 3, 3, 2, 2, 3, 3, 2, 0, 1,
       0, 2, 3, 2, 3, 0, 2, 3, 1, 0, 2, 2, 0, 1, 2, 2, 0, 3, 3, 2, 3, 0,
       2, 0, 2, 2, 3, 3])
```

In [37]:
```
### accuracy for linear svm is 96.5%
linear_ac
```

Out[37]:
```
0.965
```